ADHIYAMAAN COLLEGE OF ENGINEERING (AUTONOMOUS)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

PROFESSIONAL READINESS FOR INNOVATION, EMPLOYABILITY AND ENTREPRENEURSHIP

TOPIC: CAR RESALE VALUE PREDICATION

Team ID-PNT2022TMID08070

TEAM LEADER :

AJAYSIVAN

TEAM MEMBERS :

GNANENDRA PRASAD
DRAVIDKUMAR
ALFREDSAMSTEPHEN
ARAVINDNANDHA

Customer Segmentation Analysis

Problem Statement:-

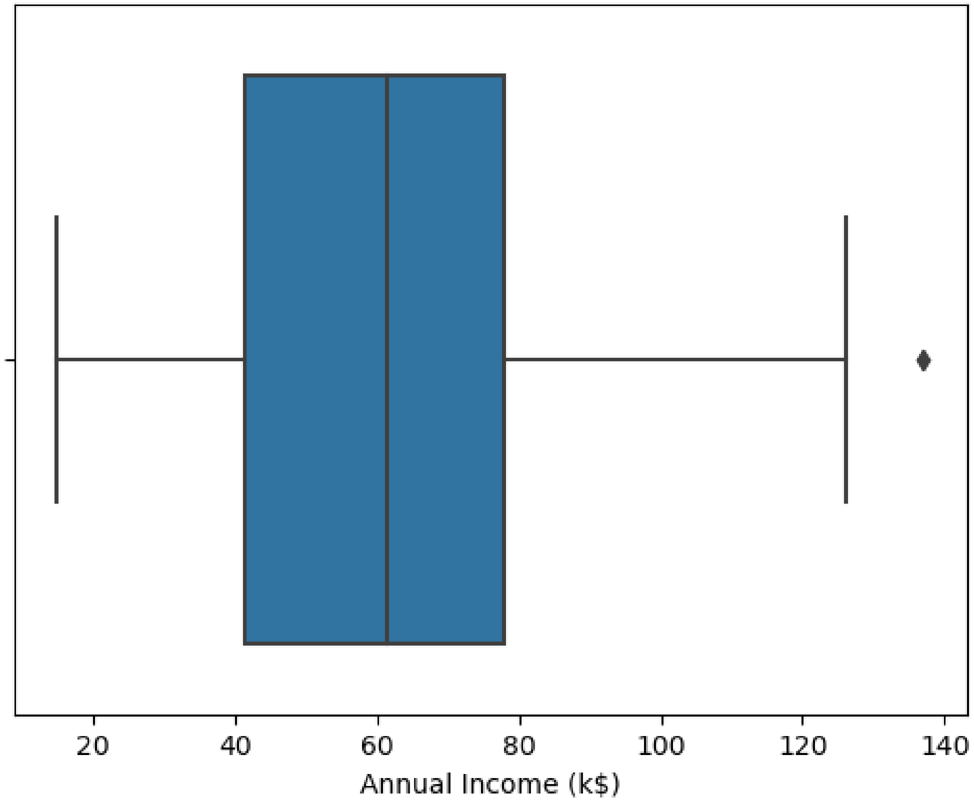
You own the mall and want to understand the customers who can quickly converge [Target Customers] so that the insight can be given to the marketing team and plan the strategy accordingly.

Clustering the data and performing classification algorithms
1. Perform Below Visualizations.
Import Libraries

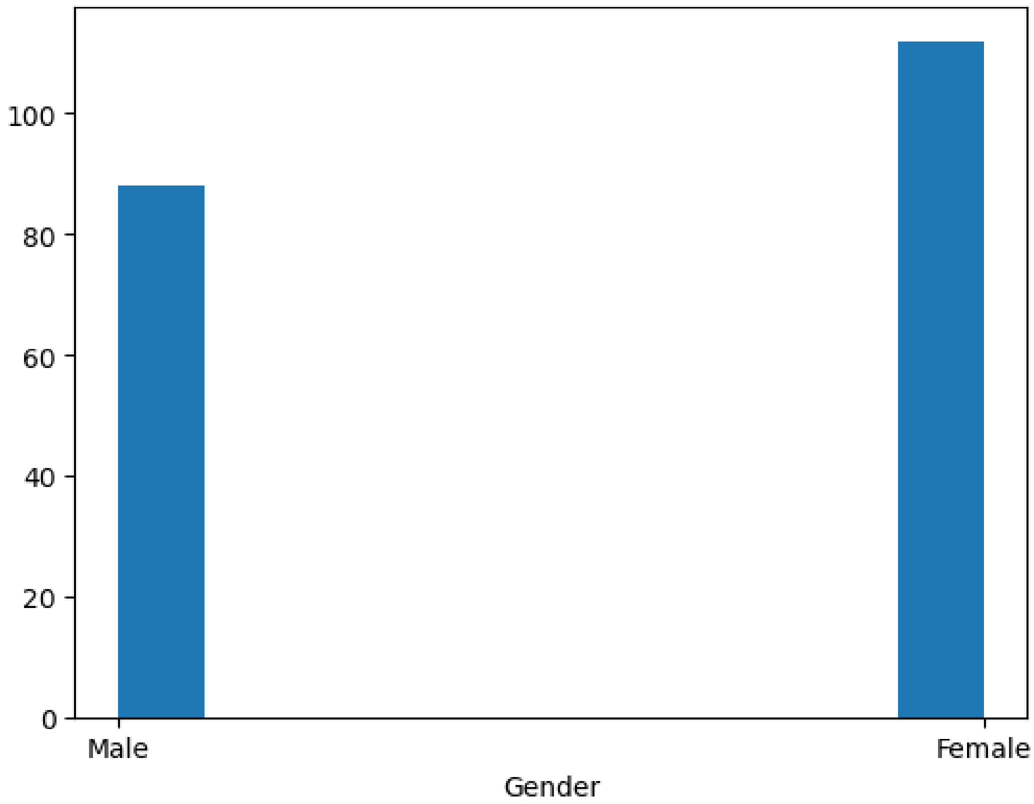
Coding:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.cluster import KMeans
from sklearn.metrics import confusion_matrix,classification_report, accuracy_score
# Import Dataset
data = pd.read_csv('Mall_Customers.csv')
data
data.info()
```

Univariate Analysis

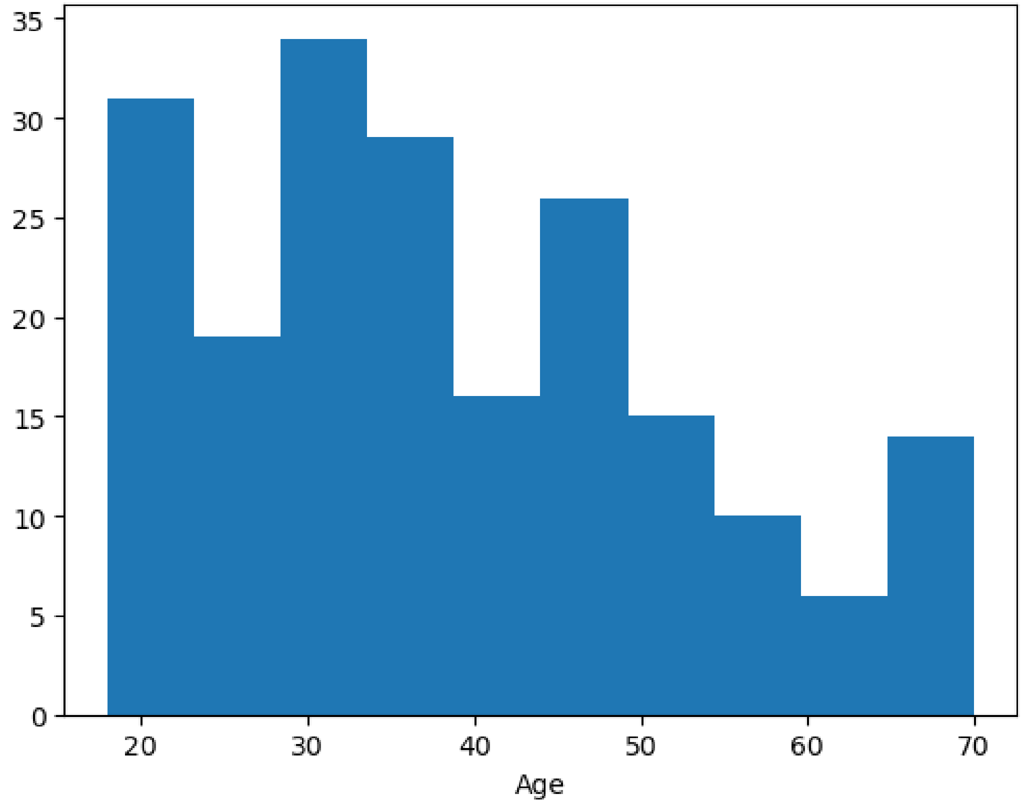



```
plt.hist(data['Gender']);
plt.xlabel('Gender');
```



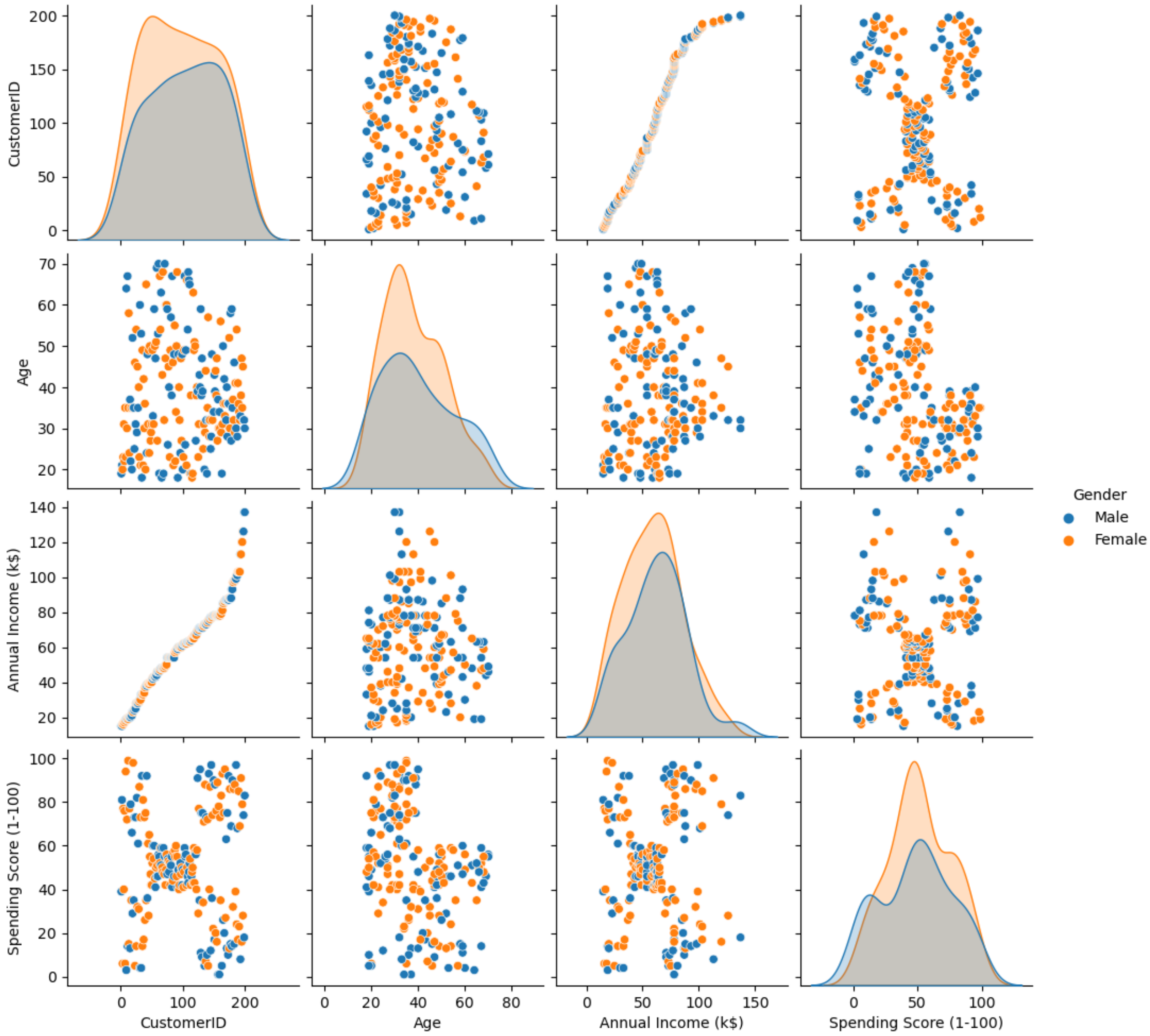

```
plt.hist(data['Age']);
plt.xlabel('Age');
```

```
plt.hist(data['Annual Income (k$)']);
plt.xlabel('Annual Income (k$)');
```




```
sns.boxplot(x=data['Annual Income (k$)'])
plt.xlabel('Annual Income (k$)');
plt.hist(data['Spending Score (1-100)']);
plt.xlabel('Spending Score (1-100)');
# Bivariate Analysis
plt.figure(figsize=(10, 6))
sns.lineplot(x=data["Age"], y=data["Annual Income (k$)"]);
plt.xlabel('Age');
plt.ylabel('Annual Income (k$)');
plt.figure(figsize=(10, 6))
sns.lineplot(x=data["Age"], y=data["Spending Score (1-100)"]);
plt.xlabel('Age');
plt.ylabel('Spending Score (1-100)');
```

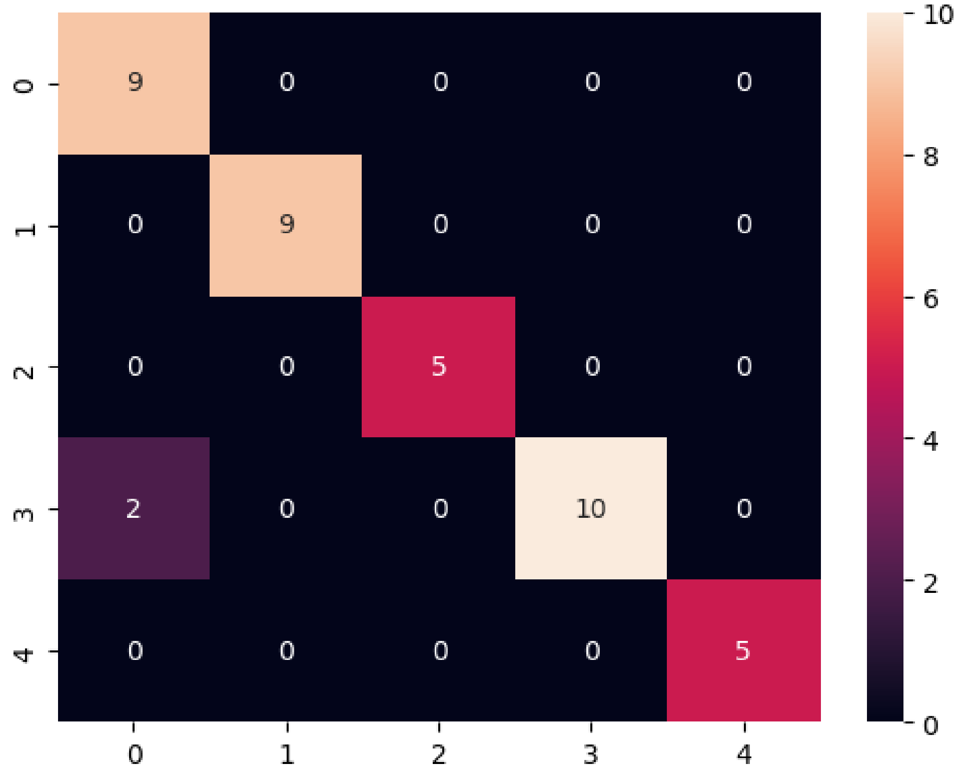
Multi-variate Analysis



sns.pairplot(data, hue='Gender');
plt.figure(figsize=(10, 6));
sns.heatmap(data.corr(), annot=True);
# Descriptive Statistics
data.describe()
data.skew()
data.kurt()
data.var()
# Handling Missing Values
data.isna().sum()
# Outlier Handling
numeric_cols = ['Age', 'Annual Income (k$)', 'Spending Score (1-100)']

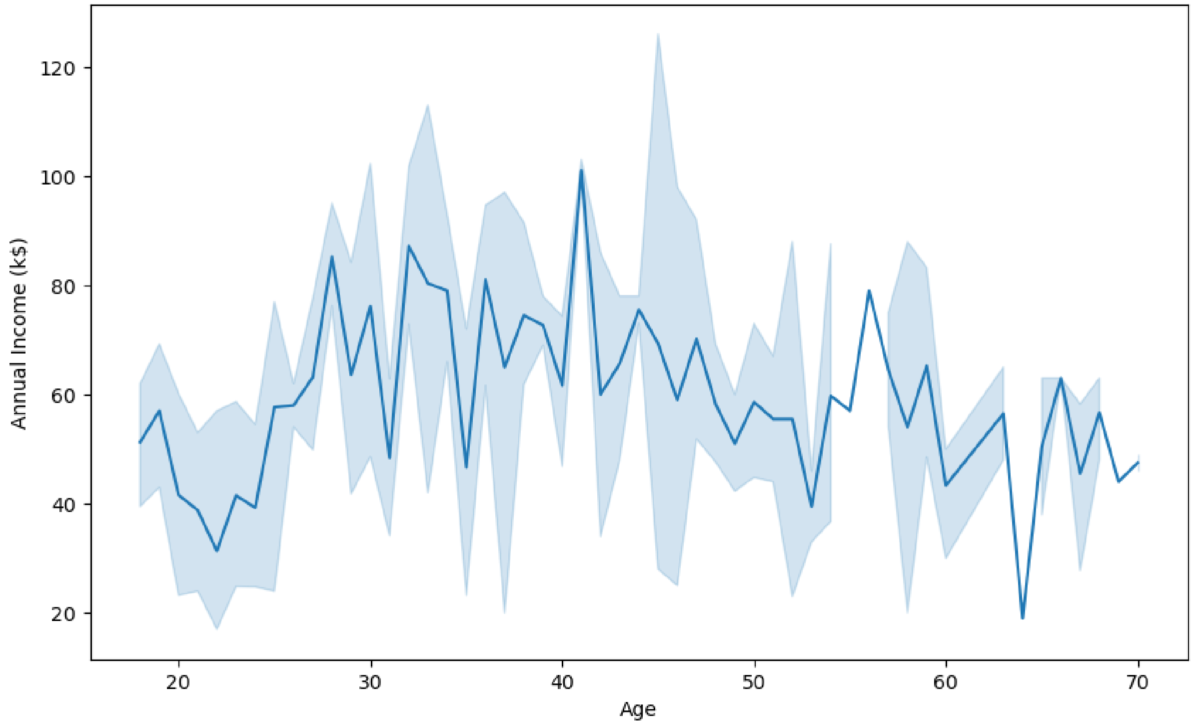
```
def boxplots(cols):

  fig, axes = plt.subplots(3, 1, figsize=(15, 20))

  t=0
  for i in range(3):
    sns.boxplot(ax=axes[i], data=data, x=cols[t])
```

```
        t+=1

    plt.show()
def Flooring_outlier(col):
    Q1 = data[col].quantile(0.25)
    Q3 = data[col].quantile(0.75)
    IQR = Q3 - Q1
    whisker_width = 1.5
    lower_whisker = Q1 -(whisker_width*IQR)
    upper_whisker = Q3 + (whisker_width*IQR)

data[col]=np.where(data[col]>upper_whisker,upper_whisker,np.where(data[col]<lower_whisker,lo
wer_whisker,data[col]))
print('Before Outliers Handling')
print('='*100)
boxplots(numeric_cols)
```




```
for col in numeric_cols:
    Flooring_outlier(col)
print('\n\n\nAfter Outliers Handling')
print('='*100)
boxplots(numeric_cols)
# Encode Categorical Columns
data = pd.get_dummies(data, columns = ['Gender'])
data
# Standard Scaling
data = data.drop(['CustomerID'], axis=1)
data
```

```
cols = data.columns
```




```
cols
scaler = StandardScaler()
sc = scaler.fit_transform(data[['Age', 'Annual Income (k$)', 'Spending Score (1-100)']])
sc
data[['Age', 'Annual Income (k$)', 'Spending Score (1-100)']] = sc
data
# Clustering
TWSS = []
k = list(range(2,13))

for i in k:
    kmeans = KMeans(n_clusters = i , init = 'k-means++')
    kmeans.fit(data)
    TWSS.append(kmeans.inertia_)

TWSS
plt.plot(k, TWSS, 'ro--')
plt.xlabel('# Clusters')
plt.ylabel('TWSS')
model = KMeans(n_clusters = 5)
model.fit(data)
# Add the Cluster data with Primary dataset
mb = pd.Series(model.labels_)
data['Cluster'] = mb
data
data[['Age', 'Annual Income (k$)', 'Spending Score (1-100)']] = scaler.inverse_transform(data[['Age',
'Annual Income (k$)', 'Spending Score (1-100)']])
data
mb=pd.Series(model.labels_)
data
```

```
# Split Data Into Dependent & Independent Features
X=data.drop('Cluster',axis=1)
Y=data['Cluster']
X, Y
# Split the data into Training And Testing Data
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_state=42)
X_train.shape, X_test.shape, Y_train.shape, Y_test.shape
# Train Model & Evaluate
model=DecisionTreeClassifier()
model.fit(X_train,Y_train)
# Evaluate
model.score(X_train, Y_train)
model.score(X_test, Y_test)
Y_pred = model.predict(X_test)
accuracy_score(Y_pred, Y_test)
print(classification_report(Y_pred, Y_test))
cm = confusion_matrix(Y_pred, Y_test)
cm
sns.heatmap(cm, annot=True);
```